

In the Specification:

[0035] As described in detail below, the add-compare-select (ACS) instruction specifies a destination register, a first pair of source registers, a second pair of source registers and several operations. During an add operation specified by the ACS instruction 308, one register of the first pair of source registers is added to one register of the second pair of source registers, thereby forming a first sum, and the other register of the first pair of source registers is added to the other register of the second pair of source registers, thereby forming a second sum. A minimum of the first and second sums is stored in the destination register.

[0046] The execution unit 406 is used to perform operations specified by instructions (and corresponding decoded instructions). In the embodiment of Fig. 4, the execution unit 406 includes an arithmetic logic unit (ALU) 412 and a multiply/accumulate units unit (MAU) 416. The ALU 412 includes 2 independent arithmetic logic units (ALUs): a 16-bit ALU0 labeled 414A, and a 16-bit ALU1 labeled 414B. The MAU 416 includes 2 independent multiply/accumulate units (MACs): MAC0 labeled 418A and MAC1 labeled 418B. The MAU 416 also includes a 40-bit arithmetic logic unit (ALU) 420.

[0064] Fig. 7A is a diagram of another embodiment of the add-compare-select (ACS) instruction 308 of Fig. 3. In the embodiment of Fig. 7A, the add-compare-select (ACS) instruction 3089 308 includes an opcode field 700, a destination register field 702, a source register 1 field 704, and a source register 2 field 706. The opcode field 700 contains a value identifying the instruction as an add-compare-select (ACS) instruction and specifying the particular Add compare select add-compare-select (ACS) instruction format of Fig. 7A.

[0086] With the Viterbi algorithm, however, only the most likely paths in the trellis diagram “survive” at each stage. As a result, at ~~most~~ most M paths survive, regardless of the number of stages. At each stage, a cost metric is used to select a “survivor” path from among the two incoming paths to each state (i.e., node). As described above, a “branch” is a transition between states (i.e., nodes). In Fig. 2, arrows between the states or nodes represent transitions or branches between the states or nodes.